PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-265402
(43)Date of publication of application : 07.10.1997

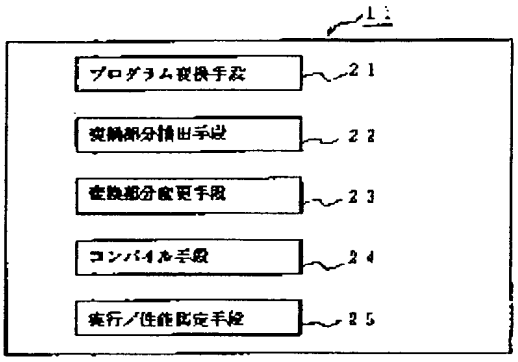
(51)Int.Cl. G06F 9/45

(21)Application number : 08-073011
(22)Date of filing : 28.03.1996

(71)Applicant : HITACHI SOFTWARE ENG CO LTD
(72)Inventor : NAKASHIGE AKIRA

(54) METHOD AND DEVICE FOR SUPPORTING PROGRAM CONVERSION

(57)Abstract:
PROBLEM TO BE SOLVED: To prevent the explosive increase of the size of a program after conversion and to improve the performance of the program after the conversion by providing a means for individually specifying a part for performing program conversion.
SOLUTION: From a variable provided with a known value in the program and the variable capable of calculating a value other than that, the part of the program capable of being calculated beforehand is extracted by a conversion part extraction means 22. In the case that the request of the program conversion is obtained by using an input device such as a mouse or the like, only the part influenced by the specified contents is similarly extracted by the conversion part extraction means 22 and displayed. In the case that the request for not performing the program conversion even in the extracted limited influence part is present, a conversion unnecessary part is cancelled by a conversion part change means 23. Further, in the case that the request for continuously performing the program conversion is not present, the execution performance of the program before and after the program conversion is measured and compared by a compilation means 24 and an execution/performance measurement means 25.



EGAL STATUS
Date of request for examination]
Date of sending the examiner's decision of rejection]
Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
Date of final disposal for application]
Patent number]
Date of registration]
Number of appeal against examiner's decision of rejection]
Date of requesting appeal against examiner's decision of rejection]
Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－265402

(43)公開日　平成９年(1997)10月７日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/45			Ｇ０６Ｆ　9/44	３２２Ｆ	
				３２２Ｚ	

審査請求　未請求　請求項の数２　ＯＬ　（全 12 頁）

(21)出願番号　特願平8－73011

(22)出願日　平成８年(1996)３月28日

(71)出願人　000233055
日立ソフトウエアエンジニアリング株式会社
神奈川県横浜市中区尾上町６丁目81番地

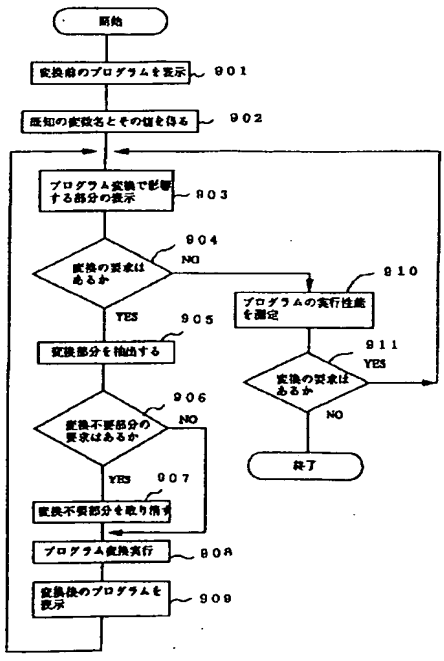
(72)発明者　中重　亮
神奈川県横浜市中区尾上町６丁目81番地
日立ソフトウェアエンジニアリング株式会社内

(74)代理人　弁理士　秋田　収喜

(54)【発明の名称】　プログラム変換支援方法および装置

(57)【要約】
【課題】　コンパイラによる最適化処理が効果的に作用するように、プログラム変換を支援し、変換後のプログラムの性能を向上させること。
【解決手段】　部分計算手法等のプログラム変換方法において、プログラム変換を行う部分を個別に選択する手段を設け、変換対象として指定した部分のみを変換させる。

図９



【特許請求の範囲】
【請求項１】　プログラムへの入力データの一部または全部を用いてそのプログラムの高速化を行うプログラム変換支援方法であって、
プログラム中のそれぞれの部分に対して、プログラム変換を行うか否かを指定した後、その指定された部分のみプログラム変換を実施させることを特徴とするプログラム変換支援方法。
【請求項２】　プログラムへの入力データの一部または全部を用いてそのプログラムの高速化を行うプログラム変換支援装置であって、
プログラム中のそれぞれの部分に対して、プログラム変換を行うか否かを指定する手段と、該手段によって指定された部分に対してプログラム変換を行う変換手段とを有することを特徴とするプログラム変換支援装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、個々の計算機プログラムをその計算結果を変えずに高速な実行を可能とする新しいプログラムへと変換するプログラム変換支援方法および装置に関する。
【０００２】
【従来の技術】従来のプログラム変換による高速化技術として、例えば特開昭６２－２６４３３３号公報に公開されたプログラム最適化処理方法がある。これは、プログラム中の関数呼び出し部分で、インラインコードに展開するかどうかを指定して実行性能の向上を図るものであるが、プログラムへの入力データの一部または全部を既知として利用したり、定数伝播などのプログラム変換の手法を総合的に用いるものではない。
【０００３】総合的にプログラム変換の手法を用いる技術として最も実用に近いものとしては、予めプログラムの変換すべき部分を洗い出してプログラム中に注記しておく部分計算手法がある。
【０００４】これは、プログラムへの入力データの一部または全部を用いて、そのプログラムを特定の計算が高速に実行できる新しいプログラムへと変換するものである。
【０００５】プログラムへの実際の作用は、入力データを含む定数の伝播、関数や式のインラインコード展開、繰り返し処理の展開、部分式の計算、条件判定の計算による分岐の選択、関数のクローニングなどである。
【０００６】この手法の具体的なアルゴリズムは、Jones、Gomard、Sestoft著 "Partial Evaluation and Automatic ProgramGeneration"、 Prentice Hall に記載されている。
【０００７】
【発明が解決しようとする課題】ところが、上記の部分計算手法はプログラムの変換技術としては強力なもので、理論上は高速なプログラムへと変換されるはずであるが、実際には必ずしも期待通りに性能が向上するとは限らないことがプロトタイプでの実験で確認されている。
【０００８】これは主として変換後のプログラムの大きさが増大する傾向にあることによる。このため、コンパイラによる最適化処理が効果的に作用しなくなっているためである。
【０００９】本発明の目的は、上記の部分計算手法によるプログラム変換を効果的に実行させ、変換後のプログラムの性能を向上させることが可能になるプログラム変換支援装置を提供することにある。
【００１０】
【課題を解決するための手段】上記の部分計算手法では一括してプログラム変換を総合的に実行するのであるが、本発明では、上記目的を達成するために、プログラム変換を行う部分を個別に指定する（または選択する）手段を設け、指定された部分のみプログラム変換を行い、変換後のプログラムの大きさの爆発的増大を防ぎ、コンパイラの最適化処理が有効に働く範囲に収めるようにしたものである。
【００１１】
【発明の実施の形態】以下、図を用いて本発明の実施形態について説明する。
【００１２】図１は、本発明のプログラム変換支援方法を実現する情報処理システムの実施形態を示すブロック構成図である。
【００１３】本実施形態の装置は、プログラムの編集・変換を行う中央処理装置１１、プログラムおよびそのプログラムへの入力データ、プログラム変換を実行する部分の指定などを入力あるいは指定する手段であるキーボード１２およびマウス１３、変換前と変換後のプログラム、およびプログラム変換に必要な補助データを表示する手段であるグラフィックディスプレイ１４、変換前と後のプログラムおよびプログラム変換に必要な補助データを格納する手段であるディスク１５から構成されている。
【００１４】図２は中央処理装置１１の詳細を示す機能ブロック図である。
【００１５】中央処理装置１１は、入力データの一部または全部を用いて特定目的のプログラムを生成するプログラム変換手段２１と、プログラム変換を適用した際に影響を受ける箇所を洗い出すための変換部分抽出手段２２と、前記影響箇所をプログラム変換による性能向上効果が大きくなるように変更するための変換部分変更手段２３と、プログラムを機械語へと翻訳するコンパイル手段２４と、プログラムの実行によりプログラム変換の効果を測る性能測定手段２５とによって構成される。
【００１６】図３はキーボード等によって本装置内のディスクに保存されている変換前のプログラムである。プログラムはＣ言語で記述してある。しかし、本発明はプ

ログラミング言語がＣ言語である必要はない。

【００１７】図３のプログラムは、「関数 pgm 」が２つの引数 x と y を通じてデータを受け取り、副プログラムの「関数 subpgm 」を呼び出して、大域変数の配列 a[]に計算結果を貯め込んでゆくものである。

【００１８】プログラム中の関数 cos はライブラリにある数学関数で、cosは三角関数のうちの余弦関数である。

【００１９】図４は、図３のプログラムと同じものであるが、「関数 pgm 」において引数x に受け渡されるデータが既知であると仮定してプログラム変換を施した場合、影響を受ける部分の一部を下線で表示している。

【００２０】この影響を受ける部分の洗い出しは、中央処理装置１１の変換部分抽出手段２２による。影響を受ける部分は、関数 pgm 中では変数 x と x で受け渡されるデータに独立に計算可能な局所変数 i、さらに x と i から計算可能なプログラム中の部分式である。また、呼び出される「関数 subpgm 」中では pgm の x の値のみで計算できる結果が引数 x を介して直接使われている箇所である。

【００２１】図４では、簡単のため、関数pgm 中の x と i、 subpgm中の x の他は、特に注目すべき部分のみに下線を付けて示している。すなわち、繰り返しを表す for文とライブラリ関数 pow の呼び出し部分である。

【００２２】図５は、図４の中の下線部分を中央処理装置１１のプログラム変換手段２１によって変換した結果得られたプログラムを示したものである。

【００２３】図５においては、関数 pgm の引数 x と局所変数 i に関係する部分が計算されて新しプログラムに置き換わっている。具体的には、関数 pgm 中の引数 x がなくなり、繰り返しの計算を行う for 文が局所変数 i の値にしたがって展開され、i 自身はなくなっている。また、関数 subpgm でも引数 x のデータに直接依存する引数がなくなって計算値に置き換わっている。

【００２４】このプログラム変換の際には 関数 pgm の引数 x の値が既知で、「２」と仮定している。

【００２５】このような仮定は、プログラムを実行する際に引数 x が特定の値「２」のときだけである場合、値「２」の場合の実行頻度が非常に大きい場合、値「２」の場合には特に実行性能が要求される場合、などで有効である。

【００２６】図５以降のプログラム変換は上記のような場合に実行されるものである。

【００２７】さらに、次のプログラム変換の段階として、図５中に下線が表示してある。

【００２８】これらは関数 pgm 中の局所変数 m に依存する部分について、変換部分抽出手段２２によって洗い出されたものである。この図５では、変数 m の値は関数 subpgm 中にも影響を与えることが分かる。

【００２９】図６は、図５の下線部分についてプログラム変換手段２１によって変換して得られたプログラムである。ここでは、関数 pgm 内の局所変数 m の値「２」、「４」、「１６」、「２５６」にしたがって関数 subpgm のクローンが４個作られ、それぞれの関数内で m の値が定数伝播によって使われている。

【００３０】これら４個のクローン関数 subpgm_2, subpgm_4,subpgm_16, subpgm_256 は関数 pgm からそれぞれ１回ずつ呼ばれている。

【００３１】さらに、次のプログラム変換の段階として、図６中に下線が表示してある。これらは関数 subpgm_2, subpgm_4, subpgm_16 中の局所変数 j に依存する部分について、変換部分抽出手段２２によって洗い出されたものである。

【００３２】この図６では繰り返しを表す for 文 にも下線を施してある。

【００３３】ここで、関数 subpgm_256 中の局所変数 j については、プログラム変換後のプログラムの大きさの増大を考慮して、次のプログラム変換の対象から外してある。

【００３４】図７は、図６の下線部分についてプログラム変換手段２１によって変換して得られたプログラムである。

【００３５】関数 subpgm_2, subpgm_4, subpgm_16 内のそれぞれの局所変数 j がなくなり、繰り返しを表すfor文が展開され、j の値にしたがってライブラリ関数 cos の引数部分の値が計算されている。

【００３６】さらに、次のプログラム変換の段階として、図７中に下線が表示してある。これは関数 subpgm_2, subpgm_4, subpgm_16 中のライブラリ関数 cos の呼び出し部分で、変換部分抽出手段２２によって洗い出されたものである。

【００３７】図８は、上記図７の下線部分についてプログラム変換手段２１によって変換して得られたプログラムである。

【００３８】関数 subpgm_2, subpgm_4, subpgm_16 中のライブラリ関数 cos の呼び出し部分がそれぞれの計算値である定数に置き換わっている。

【００３９】以上のような図３から図８までのプログラム例で、プログラム変換にも様々な段階がとれることが分かる。

【００４０】プログラム中の一部分の実行内容を、特にプログラムへの入力データの一部または全部を使って前もって計算しておくと、変換後のプログラムは計算すべき内容が減るために実行性能が向上するはずである。しかし、上記の例で見たように、変換後のプログラムは増大する可能性があることが分かる。

【００４１】特に、プログラムのサイズが爆発的に増大する場合は、一度に一括してプログラム変換を施しても最も高速に実行するプログラムが得られるとは限らない。

【００４２】したがって、上記のようなプログラム変換を一部分ずつ、段階を追って施すのが最も効果的であることが分かる。本発明は、これを可能にするための支援装置である。

【００４３】図９は、本発明のプログラム変換支援装置におけるプログラム変換の過程を表すフローチャートである。

【００４４】プログラム変換支援装置は、まず変換前のプログラムをグラフィクスディスプレイ１４等の表示装置に表示する(ステップ９０１)。このプログラムはキーボード１２などの入力装置等によりディスク１５等の保存装置に蓄えられたものである。

【００４５】次に、プログラム中で既知の値を持つ変数(或は関数の引数)があるときは、変数名と、その値をキーボード１２等の入力装置から得る(ステップ９０２)。

【００４６】この既知の値を持つ変数と、それ以外の値が計算可能な変数とから、予め計算しておけるプログラムの部分を変換部分抽出手段２２により洗い出し、ディスプレイ１２等の表示装置に表示する(ステップ９０３)。

【００４７】ここで、プログラム変換の要求がマウス１３等の入力装置を用いて得られた場合(ステップ９０４)、その指定内容が影響する部分のみについて、上記ステップ９０２と同様に変換部分抽出手段２２により洗い出し、表示を行う(ステップ９０５)。

【００４８】このプログラム変換の要求は、具体的には上記ステップ９０３で表示した影響部分の中で、引数名、変数名、関数名などを指定することによる。

【００４９】更にステップ９０５で洗い出した、限定された影響部分の中でもプログラム変換を行わないという要求がある場合(ステップ９０６)、変換部分変更手段２３によって変換不要部分を取り消す(ステップ９０７)。この要求も上記のステップ９０４での要求と同様に指定されるものである。

【００５０】次に、プログラム変換手段２１によってプログラム変換を実行し(ステップ９０８)、変換後のプログラムを表示する(ステップ９０９)。

【００５１】さらに、プログラム変換を続けて行う要求に備えてステップ９０３に戻り、変換後のプログラムに対して変換部分抽出手段２２を用いて、次の変換で影響する部分を洗い出して表示する。

【００５２】ステップ９０４でプログラム変換の要求がない場合は、コンパイル手段２４および実行／性能測定手段２５によって、プログラム変換の前と後のプログラムの実行性能を測定し、比較する(ステップ９１０)。

【００５３】この後、さらにプログラム変換の要求がある場合は（ステップ９１１）、ステップ９０３に戻り、ない場合はプログラム変換を終了する。

【００５４】以上のように、本実施形態によれば、プログラム変換を行う部分を個別に順次選択することにより、コンパイラの最適化処理が有効に働くような段階を選ぶことができ、その結果として、変換後のプログラムの大きさの爆発的増大を防ぐことが可能となる。このため、部分計算手法等のプログラム変換手法を、プログラムの実行性能の向上に効果的に適用することができ、変換後のプログラムの性能を向上させることができる。

【００５５】

【発明の効果】以上の説明から明らかなように、本発明によれば、プログラム変換を行う部分を個別に選択する手段を設け、コンパイラの最適化処理が有効に働くような段階を選んでプログラム変換を行うように支援することにより、変換後のプログラムの大きさの爆発的増大を防ぐことが可能となり、変換後のプログラムの性能を向上させることができる。

【図面の簡単な説明】

【図１】本発明に係るプログラム変換支援装置の実施形態を示すブロック構成図である。

【図２】図１における中央処理装置の詳細構成を示す機能ブロック図である。

【図３】プログラム変換を行う前のプログラムの一例を示す図である。

【図４】図３のプログラムについて、プログラム変換を行うことが可能な一部分に下線を付けた例を示す図である。

【図５】図４のプログラムについて、プログラム変換を実行した例を示す図である。

【図６】図５のプログラムについて、更にプログラム変換を実行した例を示す図である。

【図７】図６のプログラムについて、更にプログラム変換を実行した例を示す図である。

【図８】図７のプログラムについて、更にプログラム変換を実行した例を示す図である。

【図９】本発明のプログラム変換支援過程を示すフローチャートである。

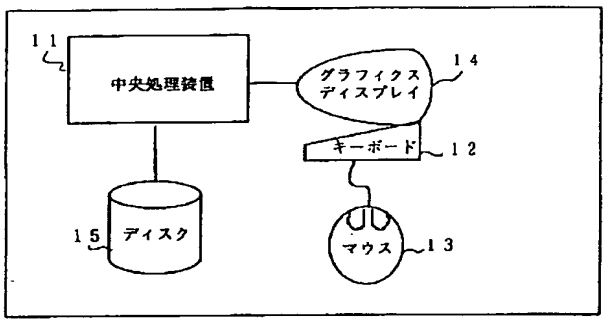
【符合の説明】

１１…中央処理装置、１２…キーボード、１３…マウス、１４…グラフィクスディスプレイ、１５…ディスク、２１…プログラム変換手段、２２…変換部分抽出手段、２３…変換部分変更手段、２４…コンパイル手段、２５…実行／性能測定手段。

【図1】

図1

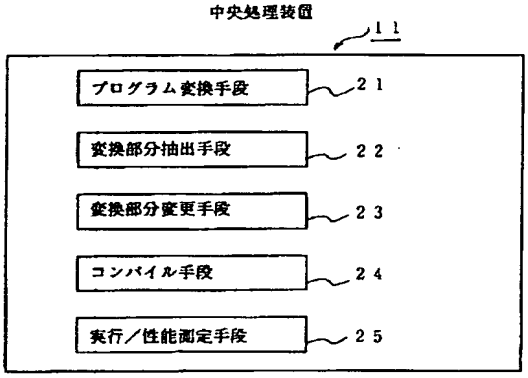
プログラム変換支援装置



【図2】

図2

中央処理装置



【図3】

図3

変換前のプログラム

```
double a[512];

void subpgm(m, b, x, y)
int m, b, x;
double y;
{
  int j;

  for(j=1; j<=m; j++) {
      a[b+j] = y * cos(j+x);
    }
}

void pgm(x, y)
int x;
double y;
{
  int b = 0;
  int i, m;

  for(i=0; i<=3; i++) {
      m = (int)pow(x, pow(x, i));
      subpgm(m, b, x, y);
      b = b + m;
    }
}
```

【図4】

図4

変換可能な部分に註記したプログラム

```
double a[512];

void subpgm(m, b, x, y)
int m, b, x;
double y;
{
  int j;

  for(j=1; j<=m; j++) {
      a[b+j] = y * cos(j+x);
    }
}

void pgm(x, y)
int x;
double y;
{
  int b = 0;
  int i, m;

  for(i=0; i<=3; i++) {
      m = (int)pow(x, pow(x, i));
      subpgm(m, b, x, y);
      b = b + m;
    }
}
```

【図5】

図5

変換後のプログラム(1)

```
double a[512];

void subpgm(m, b, y)
int m, b;
double y;
{
  int j;
  for(j=1; j<=m; j++) {
      a[b+j] = y * cos(j+2);
    }
}

void pgm(y)
double y;
{
  int b = 0;
  int m;

  m = 2;
  subpgm(m, b, y);
  b = b + 2;
  m = 4;
  subpgm(m, b, y);
  b = b + 4;
  m = 16;
  subpgm(m, b, y);
  b = b + 16;
  m = 256;
  subpgm(m, b, y);
  b = b + 256;
}
```

【図6】

図6

変換後のプログラム（2）

```
double a[512];

void subpgm_2(b, y)
int b;
double y;
{
  int j;
  for(j=1; j<=2; j++) {
      a[b+j] = y * cos(j+2);
    }
}

void subpgm_4(b, y)
int b;
double y;
{
  int j;
  for(j=1; j<=4; j++) {
      a[b+j] = y * cos(j+2);
    }
}

void subpgm_16(b, y)
int b;
double y;
{
  int j;
  for(j=1; j<=16; j++) {
      a[b+j] = y * cos(j+2);
    }
}

void subpgm_256(b, y)
int b;
double y;
{
  int j;
  for(j=1; j<=256; j++) {
      a[b+j] = y * cos(j+2);
    }
}
```

```
void pgm(y)
double y;
{
  int b = 0;
  subpgm_2(b, y);
  b = b + 2;
  subpgm_4(b, y);
  b = b + 4;
  subpgm_16(b, y);
  b = b + 16;
  subpgm_256(b, y);
  b = b + 256;
}
```

【図7】

図7

変換後のプログラム（3）

```
double a[512];

void subpgm_2(b, y)
int b;
double y;
{
  a[b+1] = y * cos(3);
  a[b+2] = y * cos(4);
}

void subpgm_4(b, y)
int b;
double y;
{
  a[b+1] = y * cos(3);
  a[b+2] = y * cos(4);
  a[b+3] = y * cos(5);
  a[b+4] = y * cos(6);
}

void subpgm_16(b, y)
int b;
double y;
{
  a[b+1] = y * cos(3);
  a[b+2] = y * cos(4);
  a[b+3] = y * cos(5);
  a[b+4] = y * cos(6);
  a[b+5] = y * cos(7);
  a[b+6] = y * cos(8);
  a[b+7] = y * cos(9);
  a[b+8] = y * cos(10);
  a[b+9] = y * cos(11);
  a[b+10] = y * cos(12);
  a[b+11] = y * cos(13);
  a[b+12] = y * cos(14);
  a[b+13] = y * cos(15);
  a[b+14] = y * cos(16);
  a[b+15] = y * cos(17);
  a[b+16] = y * cos(18);
}

void subpgm_256(b, y)
int b;
double y;
{
  int j;
  for(j=1; j<=256; j++) {
      a[b+j] = y * cos(j+2);
    }
}

void pgm(y)
double y;
{
  int b = 0;

  subpgm_2(b, y);
  b = b + 2;
  subpgm_4(b, y);
  b = b + 4;
  subpgm_16(b, y);
  b = b + 16;
  subpgm_256(b, y);
  b = b + 256;
}
```

【図8】

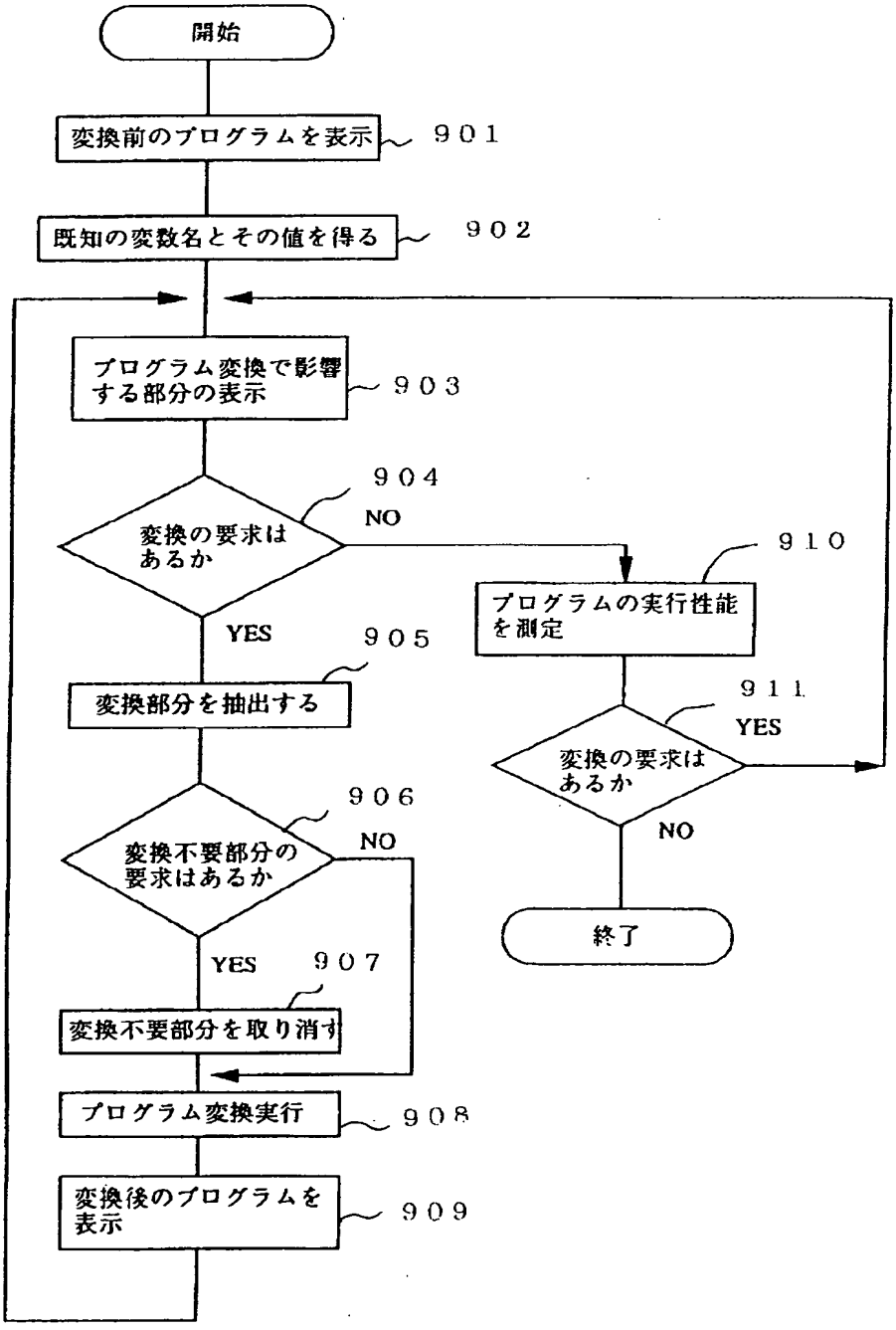
図8

変換後のプログラム（4）

```
double a[512];

void subpgm_2(b, y)
int b;
double y;
{
  a[b+1] = y * -9.899925e-01;
  a[b+2] = y * -6.536436e-01;
}

void subpgm_4(b, y)
int b;
double y;
{
  a[b+1] = y * -9.899925e-01;
  a[b+2] = y * -6.536436e-01;
  a[b+3] = y * 2.836622e-01;
  a[b+4] = y * 9.601703e-01;
}

void subpgm_16(b, y)
int b;
double y;
{
  a[b+1] = y * -9.899925e-01;
  a[b+2] = y * -6.536436e-01;
  a[b+3] = y * 2.836622e-01;
  a[b+4] = y * 9.601703e-01;
  a[b+5] = y * 7.539023e-01;
  a[b+6] = y * -1.455000e-01;
  a[b+7] = y * -9.111303e-01;
  a[b+8] = y * -8.390715e-01;
  a[b+9] = y * 4.425698e-03;
  a[b+10] = y * 8.438540e-01;
  a[b+11] = y * 9.074468e-01;
  a[b+12] = y * 1.367372e-01;
  a[b+13] = y * -7.596879e-01;
  a[b+14] = y * -9.576595e-01;
  a[b+15] = y * -2.751633e-01;
  a[b+16] = y * 6.603167e-01;
}

void subpgm_256(b, y)
int b;
double y;
{
  int j;
  for(j=1; j<=256; j++) {
      a[b+j] = y * cos(j+2);
    }
}

void pgm(y)
double y;
{
  int b = 0;

  subpgm_2(b, y);
  b = b + 2;
  subpgm_4(b, y);
  b = b + 4;
  subpgm_16(b, y);
  b = b + 16;
  subpgm_256(b, y);
  b = b + 256;
}
```

【図9】

図9


開始
変換前のプログラムを表示
901
既知の変数名とその値を得る
902
プログラム変換で影響する部分の表示
903
904
変換の要求はあるか
NO
910
プログラムの実行性能を測定
YES
905
変換部分を抽出する
911
YES
変換の要求はあるか
906
変換不要部分の要求はあるか
NO
NO
終了
YES
907
変換不要部分を取り消す
プログラム変換実行
908
変換後のプログラムを表示
909